REMARKS

I. INTRODUCTION

In response to the Office Action dated January 24, 2008, the claims have not been amended. Claims 1-18 remain in the application. The specification has been slightly amended (as described below). Entry of these amendments, and re-consideration of the application, as amended, is requested.

II. SUMMARY OF THE INVENTION

There are essentially two sets of claims. The first set includes independent claims 1, 5, and 17. The second set includes claims 9 and 13. Each set will be addressed separately herein.

Claims 1, 5, and 17

These independent claims provide the ability to create and display a dynamic property on a per object basis. As explicitly set forth in the claims, the dynamic property is created at runtime for an object instance on a per-instance basis and is not stored with the object. Referring to claim 1, the first limitation provides for retrieving a reference to an object instance. This element provides an instance of an object/class for which the dynamic property will be created.

The second limitation retrieves a reference to a property source instance. Applicants note that the limitation provides that it is an instance of the property source (i.e., it is an instance of an object). Further the reference to this property source instance is obtained. The second limitation also provides that the reference is retrieved from "an association between the object and the property source instance". Thus, rather than retrieving a reference from a random location, it is retrieved from an entity identified as an "association" in the claims. Such an association can take a variety of forms (e.g., a mapping, a linked list, etc.). In addition, the second element provides that the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance. Accordingly, when examining this claim limitation in combination with the first limitation, the claims explicitly provide that a property source instance creates (at run time) and supplies (at run time) a new property and an initial value for the new property to/for the object instance. Thus, rather than the object instance creating a value for one of its own properties at runtime (e.g., a name for the object instance), this claim element explicitly

provides that a property itself AND a value for that property is created for one object (i.e., the object instance) by another object (i.e., the property source instance).

The third limitation provides the two references (i.e., the reference to the object instance and the reference to the property source instance) to a control. The third limitation further provides that the control is configured to retrieve the dynamic property from the property source instance and display the dynamic property in a user interface.

As can be seen by the above limitations, there are specific and detailed claim limitations that provide a precise structure and capability. More specifically, a property source instance creates a property (and a value for that property) for an object instance. Such a creation is performed dynamically during runtime.

CLAIM LIMITATION	SPECIFICATION SUPPORT
1. A computer-implemented method for displaying per-instance dynamic properties of an object comprising:	[0005]-[0006] - Page 2, line 23-page 3, line 20; [0010] - Page 4, line 22-page 5, line 7; [0034] - page 11, lines 4-14; Fig. 4.
(a) receiving a reference to an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object;	[0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12;
(b) retrieving a reference to a property source instance from an association between the object and the property source instance, wherein the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance; and	[0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;
(c) providing the reference to the object instance and the reference to the property source instance to a control, wherein the control	[0003] - Page 2, lines 12-15; [0039] - Page 12, line 22-page 13, line 8; [0042] - Page 13, line 20-page 14, line 3; [0051]-[0053] - page 17, line 14-page

is configured to:	18, line 12; FIG. 5, steps 508-516; [0054]-[0058] - Page 18, line 15-page 20, line 3; FIG. 4, item 410; [0062] Page 21, lines 8-12;
(i) retrieve the dynamic property from the property source instance; and	[0045] - Page 15, lines 4-12; FIG. 4, item 410.
(ii) display the dynamic property in a user interface.	[0045] - Page 15, lines 4-12; FIG. 4, item 410.
5. A computer-implemented system for displaying per-instance dynamic properties of an object comprising:	[0005]-[0006] - Page 2, line 23-page 3, line 20; [0010] - Page 4, line 22-page 5, line 7; [0034] - page 11, lines 4-14; Fig. 4.
(a) a computer;	[0022] - Page 7, lines 9-17; FIG. 1, 100.
(b) an application executing on the computer;	[0023] - Page 7, line 18-page 8, line 7; FIG. 1, 108;
(c) an object instance, in the application, having a dynamic property that is created at runtime for the object instance on a per-instance bases and is not stored with the object;	[0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12;
(d) a property source instance, in the application, wherein the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance;	[0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;
(e) an association, in the application, between the object and the property source instance; and	[0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;

(f) a host, in the application, configured to:	[0037] - Page 12, lines 7-16; FIG. 4, 412.
(i) retrieve a reference to the object instance;	[0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12;
(ii) retrieve a reference to the property source instance from the association; and	[0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;
(iii) provide the reference to the object instance and the reference to the property source instance to a control, wherein the control is configured to:	[0003] - Page 2, lines 12-15; [0039] - Page 12, line 22-page 13, line 8; [0042] - Page 13, line 20-page 14, line 3; [0051]-[0053] - page 17, line 14-page 18, line 12; FIG. 5, steps 508-516; [0054]-[0058] - Page 18, line 15-page 20, line 3; FIG. 4, item 410; [0062] Page 21, lines 8-12;
(1) retrieve the dynamic property from the property source instance; and	[0045] - Page 15, lines 4-12; FIG. 4, item 410.
(2) display the dynamic property in a user interface.	[0045] - Page 15, lines 4-12; FIG. 4, item 410.
17. A computer-implemented system for displaying per-instance dynamic properties of an object comprising:	[0005]-[0006] - Page 2, line 23-page 3, line 20; [0010] - Page 4, line 22-page 5, line 7; [0034] - page 11, lines 4-14; Fig. 4.
(a) an object instance of a class, wherein:	[0032] - Page 10, lines 13-18.

(i) an initial value for one or more static properties of the class are assigned at run time; and	[0044] - Page 14, line 14- page 15, line 3
(ii) the object instance has a dynamic property and an initial value of the dynamic property that are both generated and supplied, by a property source instance, at runtime for the object instance, on a per-instance basis and are not stored with the object;	[0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12;
(b) an association between either:	[0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;
(i) the object instance and the property source instance; or	[0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;
(ii) the class and the property source instance; and	[0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;
c) a user interface component that displays a collection of properties of the object instance including the one or more static properties and the dynamic property on a display device, wherein the user interface component is	[0010] - Page 4, line 22-page 5, line 7; [0034] Page 11, line 5-14;

configured to:	
(i) retrieve a reference to the object instance;	[0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12;
(ii) retrieve the one or more static properties from the object instance;	[0044] - Page 14, line 14-page 15, line 3; FIG. 4, 410.
(iii) access the association to determine the property source instance associated with the object instance;	[0046] Page 15, lines 4-page 16, line 3; FIG. 4, 404-408; [0051] Page 17, lines 14-19;
(iv) call a method of the determined property source instance with the reference to the associated object instance;	[0034] Page 11, lines 5-14; [0061] - Page 20, line 19-page 21, line 7; FIG. 6, 602.
(v) receive the dynamic property, from the property source instance, wherein the property source instance dynamically generated the dynamic property and an initial value for the dynamic property; and	[0045] - Page 15, lines 4-12; FIG. 4, item 410; [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504;
(vi) display the static property and the dynamic property on the display device.	[0045] - Page 15, lines 4-12; FIG. 4, item 410; [0052]-[0053] - Page 17, line 20-page 18, line 12; FIG. 5, steps 512-520.

Claims 9 and 13

These independent claims are directed towards the display of a property in a property list. More specifically, once an object with a property has been retrieved, that same object provides an ActiveX control that defines a user interface for displaying and editing the property. A list of

properties is created. The ActiveX control is used to display a user interface for one of those properties in the list. Thus, multiple ActiveX controls are used to display individual properties in a property list.

CLAIM LIMITATION	SPECIFICATION SUPPORT
9. A computer-implemented method for providing a custom graphical user interface for editing a property of an object, comprising:	[0058] - Page 19, line 20-page 20, line 3; FIG. 6; [0067]- Page 23, lines 5-11; FIG. 8.
receiving a first object having a first property, wherein the first object provides a custom ActiveX control that defines a first user interface for displaying and editing the first property;	[0067]- Page 23, lines 5-11; FIG. 8, step 802.
creating a list of one or more object properties to be displayed, wherein the list includes the first property;	[0068] - Page 23, lines 12-19; FIG. 8, step 804.
instantiating the custom ActiveX control; and	[0068] - Page 23, lines 12-19; FIG. 8, step 806.
displaying the object properties in the list, wherein the display of the first property comprises the first user interface defined by the instantiated custom ActiveX control, wherein the property may be edited through the first user interface.	[0069]- Page 23, line 20-page 24, line 2; FIG. 8, step 808.
13. A system for providing a custom graphical user interface for editing a property of an object comprising:	[0058] - Page 19, line 20-page 20, line 3; FIG. 6; [0067]- Page 23, lines 5-11; FIG. 8.

(a) a computer;	[0022] - Page 7, lines 9-17; FIG. 1, 100.
(b) an application executing on the computer;	[0023] - Page 7, line 18-page 8, line 7; FIG. 1, 108;
(c) one or more objects, in the application, wherein each object has one or more object properties;	[0032]-[0033]- Page 10, line 13-page 11, line 2; Fig. 3, 300.
(d) a property inspector, in the application, configured to:	[0039] - Page 12, line 22-page 13, line 8; FIG. 4, 410.
(i) interrogate the one or more objects to discover one or more object properties to be displayed;	[0058]-[0059] - Page 19, line 20-page 20, line 11; FIG. 4, 410; FIG. 6.
(ii) create a list of the one or more object properties to be displayed; and	[0059] - Page 20, lines 4-11; FIG. 6 and FIG. 4, 410.
(iii) instantiate and host one or more property editors;	[0059] - Page 20, lines 4-11; FIG. 6 and FIG. 4, 410; [0060] - Page 20, lines 12-18;
(e) one or more property editors, in the application, wherein:	[0059] - Page 20, lines 4-11; FIG. 6 and FIG. 4, 410; [0060] - Page 20, lines 12-18.
(i) one of the property editors comprises a custom ActiveX control specified by one of the objects; and	[0061] - Page 20, line 19-page 21, line 7; FIG. 6, 602; FIG. 4, 410.
(ii) the custom ActiveX control defines a custom graphical user interface for displaying and editing one of the object properties.	[0059]-[0061] - Page 20, line 4-page 21, line 7; FIG. 6, 602.

III. SPECIFICATION OBJECTIONS

In paragraph (2) of the Office Action, the specification was objected to for using trademarks Netscape Navigator™ and Microsoft Internet Explorer™. Specifically, the Action stated that the terms should be capitalized wherever it appears and be accompanied by the generic terminology. The Action further asserted that the use of all capital letters to distinguish the trademarks was insufficient where the trademark symbol ™ did not accompany the trademarks.

Applicants respectfully disagree with and traverse the objections of the Examiner. MPEP 608.02(v) clearly provides:

> If the product to which the trademark refers is set forth in such language that its identity is clear, the examiners are authorized to permit the use of the trademark if it is distinguished from common descriptive nouns by capitalization…
>
> …
>
> Trademarks should be identified by capitalizing each letter of the mark (in the case of word or letter marks) or otherwise indicating the description of the mark (in the case of marks in the form of a symbol or device or other nontextual form).

Applicants clearly complied with all requirements of the MPEP based on the original filing of the application which set forth the trademark in all capital letters. In this regard, the requirement set forth by the Examiner to use the ™ symbol lacks any legal foundation in the MPEP or otherwise and is without merit.

Nonetheless, in the interest of expediting prosecution (and even though not required), Applicants have amended the specification to not only include the capital letters as originally filed but to include the ™ symbol as well. Such amendments should remove any objections to the specification and place the application in better condition for appeal. Accordingly, Applicants respectfully request entry of the amendments.

IV. PRIOR ART REJECTIONS

In paragraphs (6)-(7) of the Office Action, claims 1-18 remain rejected under 35 U.S.C. §103(a) as being unpatentable over Hirsch, U.S. Patent No. 6,915,301 (Hirsch). Applicants respectfully traverse these rejections.

Specifically, independent claims 1, 5, 9, 13, and 17 were rejected as follows:

> Claim 1: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:
>
> a. receiving a reference to an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object (column 11, lines 37-58);

Hirsch discloses retrieving reference to a property source instance (data source) from an association between the object and the property source instance (binding), wherein the property source instance creates and supplies to dynamic property (column 4, lines 11-34/column 11, lines 37-59), but does not explicitly disclose creating and supplying an initial value for the dynamic property for/to the object instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

c. providing the reference to the object and the reference to the property source instance to a control which is configured to: (i) retrieve the dynamic property from the property source and (ii) display the property (object inspector window) in a user interface (column 11, 37-67).

Claim 5: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object (column 11, lines 37-59);

Hirsch discloses a property source instance wherein the property source instance creates and supplies the dynamic property (column 11, lines 37-59) but does not explicitly disclose creating and supplying an initial value for the dynamic property for/to the object instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene (column 4, lines 11-34/column 11, lines 37-59). Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

c. an association between the object and the property source instance (column 3, lines 8-27);

d. a host configured to: (i) retrieve a reference to the object instance; (ii) retrieve a reference to the property source; (iii) provide the reference to the object and the reference to the property source to a control which is configured to (1) retrieve the dynamic property from the property source and (2) display the property in a user interface (column 11, lines 37-67).

Claim 9: Hirsch discloses a method and computer system for dynamic properties of software objects comprising receiving a first object having a first property wherein the first object provides a control (calendar control) that defines a first user interface for displaying and editing the first property, but does not explicitly disclose the control is an ActiveX control. However, Hirsch does disclose supporting ActiveX controls (column 12, lines 8-31). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to provide an ActiveX control to display and edit the first property. One would have been motivated to provide an ActiveX control to display and edit the first property in order to make the design more efficient.

b. creating a list of one or more object properties to be displayed, wherein the list includes the first property (column 12, lines 8-31);

c. instantiating the custom ActiveX control (column 12, lines 8-31);

d. displaying the object properties in the list, wherein the display of the first property comprises the first user interface defined by the instantiated ActiveX control, wherein the property may be edited through the first user interface (column 12, lines 8-31).

Claim 13: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. one or more objects, wherein each object has one or more object properties (column 11, lines 37-59);

b. a property inspector configured to (i) interrogate the one or more objects to discover one or more object properties to be displayed; (ii) create a list of the one or more object properties to be displayed; (iii) instantiate and host one or more property editors (column 11, lines 37-59/column 12, lines 1-21);

Hirsch discloses one or more property editors wherein: (i) one of the property editors comprises a custom control specified by one of the objects, but does not explicitly disclose the control is an ActiveX control. However, Hirsch does disclose supporting ActiveX controls (column 12, lines 8-31). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to provide an ActiveX control specified by one of the objects. One would have been motivated to provide an ActiveX control specified by one of the objects in order to make the design more efficient.

Hirsch discloses (ii) the custom ActiveX control defines a custom graphical user interface for displaying and editing one of the object properties (column 12, lines 8-31).

Claim 17: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. an object instance of a class, wherein (i) an initial value for one or more static properties of the class are assigned at run time; and (ii) the object instance has a dynamic property that is generated by a property source instance at runtime for the object instance on a per-instance basis and are not stored with the object (column 12, lines 20-58/column 11, lines 37-59). Hirsch does not explicitly disclose an initial value is generated and supplied by a property source instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

b. an association between either: (i) the object instance and the property source instance; (ii) the class and the property source instance (column 11, lines 37-61)

c. a user interface component that displays a collection of properties of the object instance including the one or more static properties and the dynamic property on a display device(column 11, lines 37-61), wherein the user interface component is configured to:

(i) retrieve a reference to the object instance (column 11, lines 49-63);

(ii) retrieve the one or more static properties from the object instance (column 11, lines 49-63);

(iii) access the association to determine the property source instance associated with the object instance (column 11, lines 37-61);

(iv) call a method of the determined property source instance with the reference to the associated object instance (column 11, lines 37-61);

(v) receive the dynamic property, from the property source instance, wherein the property source instance dynamically generated the dynamic property (column 11, lines 49-63). Hirsch does not explicitly disclose an initial value is generated and supplied by a property source instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

(vi) display the static property and the dynamic property on the display device (column 11, lines 49-63).

Applicants traverse the above rejections. There are essentially two sets of claims. The first set includes independent claims 1, 5, and 17. The second set includes claims 9 and 13. Each set will be addressed separately herein.

Claims 1, 5, and 17

The Office Actions rejects all of the claim limitations based on Hirsch. More specifically, Hirsch, Col. 11, lines 37-67 is used to reject the claims. Col. 11, lines 37-67 provides:

> Object properties consist of named attributes that define an object's appearance in terms of a functional expression. Access to these properties are provided through the Object Inspector. Their values are set at runtime based on the calculated result of the expressions. An expression may include the names of one or more data source columns which are automatically bound to a result row at runtime.
>
> Object property expressions result in one of the following base data types: Boolean, Numeric, String, Point, PointList, and Image. Derived Numeric types include Color, DateTime, Enum, Integer, and Percentage. Derived String types include FilePath (or URL) and FontName.
>
> Referring now to FIG. 2, the object inspector 30 is shown in more detail. The object inspector 30 provides two columns, an attribute name column 100 and a property column 110. The object being inspected in FIG. 2 has properties 112, 114, 116, 118, 120, 122, 124 and 126. Particularly, properties 114, 116, 118, 120, 124 and 126 are static constants. However, properties 112 and 122 are dynamic and are evaluated at run time. Due to the dynamic properties 112 and 122, to create a data driven application, the user only needs to enter the dynamic properties and code may be automatically generated. Thus, the data binding of the dynamic properties 112 and 122 is seamlessly integrated into the property sheet 30.
>
> The Object Inspector window 30 is a dockable control bar which displays an object's properties and events. The Object Inspector may be resized horizontally when docked and both vertically and horizontally when floating. The window may dock to the left or right of the world workspace when its location overlaps the docking site while being dragged.

As can be seen from this text, Hirsch completely and entirely fails to teach, describe, or suggest, explicitly or implicitly the creation of a property source instance for a property of a separate object instance. Hirsch further fails to teach, describe, or suggest the creation of a value for such a dynamic object. Applicants note that the term "dynamic" as set forth in the present claims refers to a property that is actually created and a value for that property is actually created dynamically. Hirsch does not utilize such a definition. Instead, Hirsch describes an object inspector that merely lists all of the properties of an object. Hirsch's "dynamic" property merely refers to properties 112 and 122 that are dynamic in the sense that they are evaluated at run time and are not static constants (such as properties 114, 116, 118, 120, 124, and 126). Thus, Hirsch's dynamic properties already exist for an object but are merely evaluated at run time. Such a teaching is not even remotely similar to the present claims which explicitly provide that a property source instance actually creates and

supplies a dynamic property and a value for the dynamic property at run time while both the dynamic property and value are for a separate object instance.

The Office Action bypasses such a creation of the property and value elements and asserts that Hirsch does not explicitly disclose creating and supplying an initial value for a dynamic property to an object instance. Instead, the Action states that Hirsch discloses data sources that generate values in a scene, and objects and their properties are bound to data sources in each scene. However, once again, the claims are not directed towards merely generating values in a scene or binding objects and properties to data sources in a scene. Instead, the claims explicitly provide for a property source instance creating and supplying both a dynamic property and an initial value for the dynamic property to a separate object instance. In addition, a control is used to retrieve the dynamic property from the property source instance and display the dynamic property. Nowhere in Hirsch is there even a remote reference to a control receiving both a reference to an object instance and a property source instance, and then retrieving the dynamic property from the property source instance and displaying such a property. Instead, Hirsch merely describes an object inspector window 30 that displays an object's properties and events. Hirsch fails to teach or allude to any control whatsoever or such a control receiving references to multiple object instances (i.e., a property source instance and an object instance) as claimed.

In addition to the above, the Office Action further relies on col. 4, lines 11-34 and col. 3, lines 8-27 to teach the various limitations of claim 5. Applicants note that col. 4, lines 11-34 are not relevant whatsoever to the present claim limitations. Such text merely describes the ability for a developer to build a virtual world having a scene containing information that is linked to data stored in a database. Hirsch builds data sources using a block diagramming tool which generates database queries. Hirsch's scenes are created with a drawing editor that binds data sources to the graphical elements of the scenes. However, what is wholly and completely lacking from such text is any description of the creation of a property source instance, an object instance, a property created by the property source instance for the object instance at run time, and/or a value for the property created by the property source instance for the object instance at run time (all of which are explicit claim limitations).

In addition, to teach the association claim limitation, the Office Action relies on col. 3, lines 8-27. Such text merely supports the arguments set forth above in that Hirsch's dynamic properties

are merely properties that are evaluated at run time and are not static. Such a teaching does not and cannot teach the express claim limitations relating to the creating of both a dynamic property and a value for that property at run time.

In view of the above, Applicants submit that Hirsch does not and cannot teach, disclose, or suggest the invention as claimed.

In response to the above remarks, the final Office Action responds in paragraph (5).

The response first asserts:

> Claims 1, 5, and 17: Applicant argues, "Hirsch [does not explicitly disclose] the creation of a property source instance for a property of a separate object instance." It is respectfully submitted that the disclosure of Claim 1 does not include this limitation.

Applicants respectfully disagree with and traverse such an assertion. As explained above and as explicitly claimed, a reference to an object instance is received. Further, a reference to a property source instance from an association between the object and the property source instance is retrieved. Further yet, the claims explicitly provide that the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance. As can be clearly seen, the claims separately reference both the property source instance and the object instance. The claims also explicitly provide that an association between the two exists. Such claim language clearly provides for the creation of a property source instance for a property that is separate from the object instance. In this regard, to interpret the claim language differently lacks any foundation in both the claims and the specification.

The Action further continues and provides:

> Applicant argues, "Hirsch [does not explicitly disclose] the creation of a value for such a dynamic object." It is respectfully submitted that Hirsch creates and sets values at runtime based on calculated results of expressions. The creation of a value based on calculated expressions is viewed as dynamic because the value is created at runtime. Applicant argues Hirsch's dynamic properties already exist for an object but are merely evaluated at runtime, however, the creation of a value at runtime based on a calculated expression is dynamic.

Again, the claims not only provide for the dynamic creation of the value but for the dynamic creation of the property itself. The Action continues to assert that values are created at runtime. However, there is no mention of the dynamic creation of both the property and the initial value for the property. In this regard, Applicant's assertions above reflect that Hirsch's property already exists and the value for the property is evaluated at run time. Again, the claims provide for the dynamic

creation of two (2) elements: (1) the property itself, and (2) the initial value for the property. Hirsch fails to even remotely allude to the dynamic creation of the property itself.

The Action further provides:

> Applicant argues, "Nowhere in Hirsch is there even a remote reference to a control receiving both a reference to an object instance and a property source instance, and then retrieving the dynamic property from the property source instance and displaying such a property." It is respectfully submitted that Hirsch discloses the limitation as claimed above. Specifically, Hirsch discloses an object inspector window displays the object, the object's properties (static and dynamic), and receives the object properties from a data source. The retrieval of this information by the object inspector window to display the information provides a reference between the property source and the object instances.

Applicants respectfully traverse such an assertion. As set forth in the Action itself, Hirsch's object inspector window displays the object, the object's properties, and receives the object properties from a data source. As can clearly be seen, Hirsch's data source is not an instance of an object or a property source instance as claimed. Again, the claims provide for a control that uses a reference to a property source instance to retrieve the dynamically created property. As claimed, the dynamic property is then displayed in the user interface. Hirsch, in col. 11, line 37-col. 12, line 30 describes a window that displays properties of an object with the left column containing the name of the property and the right column displaying the property's value. Values entered in the right column can be constants or can be calculated values containing functions or parameters or column names from a data source. Thus, rather than retrieving a dynamic property itself (as claimed), Hirsch merely describes the ability to retrieve a column name from a data source. Such a teaching is not even remotely similar to nor does it teach, disclose, suggest, allude to, or hint at retrieving a dynamic property itself (in addition to the dynamically created value for the property) and displaying such a property (as claimed).

Applicants question where Hirsch provides for dynamically creating both a property and a value for that property (as claimed). Applicants further question where Hirsch describes the ability to retrieve a dynamically created property from a property source instance and displaying the property.

Claims 9 and 13

These independent claims are directed towards the display of a property in a property list. More specifically, once an object with a property has been retrieved, that same object provides an

ActiveX control that defines a user interface for displaying and editing the property. A list of properties is created. The ActiveX control is used to display a user interface for one of those properties in the list. Thus, multiple ActiveX controls are used to display individual properties in a property list.

Applicants submit that Hirsch clearly fails to teach, disclose, or suggest such unique attributes as set forth in the present claim limitations. In rejecting these claims, the office Action asserts that Hirsch's calendar control teaches the invention as claimed (col. 12, lines 8-31). Col. 12, lines 8-31 describes a 2-column entry form for setting the values of a selected object's properties. The left column displays the name of the property while the right column is a read/write column that displays the property's value. The values entered can be constants or can be calculated containing functions or parameters or column names from a data source. If a property value is of an enumerated type, a drop down combo box may be displayed listing the legal values. Further, if the property is a date or time, a calendar control may be displayed beneath the property value when the field is active.

However, what should be noted is that nowhere in Hirsch is there any description that the object itself provides the custom control to display the properties of the object. Instead, Hirsch explicitly teaches a defined set of controls that are used independent of the object. In this regard, Hirsch's object does not provide the combo box that is used to display a list of enumerated properties. Instead, an object inspector provides a list of properties and determines what controls are used to edit the properties. Such controls are thus provided by the object inspector and not the object whose properties are being displayed in the tabbed sheet (as claimed). In this regard, the object inspector's properties are not being displayed in the tabbed sheet. Instead, the object inspector is used to display a different object's properties.

In contrast to Hirsch's teaching, the presently claimed invention provides that the object provides a custom control that defines a user interface for displaying and editing one of its own properties. Such a capability is wholly and complete missing, both explicitly and implicitly from Hirsch.

Further, Hirsch does not even remotely describe that the control is a custom control. Instead, Hirsch's controls are statically defined based on the type of property being displayed (e.g., if the property value is of an enumerated type, a drop-down combo box is used or if it is a date or

time, a calendar control is used). Such a control is not a custom control provided by the object and used to edit a particular property of the object at the choice of the object itself.

In response to the above arguments, the Office Action first provides:

> Claims 9 and 1 3: Applicant argues, "Nowhere in Hirsch is there any description that the object itself provides the custom control to display the properties of the object." It is respectfully submitted that Hirsch discloses the limitation as claimed above. The objects in Hirsch are selected in an object inspector window, and provide custom controls such as calendar controls when specific objects are inspected. An object that does not contain a date or time property would not provide a calendar control, giving the objects the ability to provide custom controls based on which object Is being inspected.

Applicants reassert that arguments set forth above. Namely, the present claims provide the ability for the object itself to provide a custom control to display one of its own properties. The Action merely asserts that the object inspector window provides controls to display another object's properties. Such a teaching is not what the present claims teach, disclose, nor suggest and cannot and does not render the present claims obvious. Again, Hirsch's displaying of another object's properties via a predefined set of controls is an entirely different implementation that the claimed objects displaying their own set of properties. The Action merely ignores such a distinction and provides that an object that does not contain a date or time property would not provide a calendar control. The relevance of such a statement to the present claims is lacking and ignores the differences set forth above.

The Office Action continues and provides:

> Applicant argues, "Hirsch does not even remotely describe that the control is a custom control." It is respectfully submitted that the controls provided in Hirsch are custom controls due to the fact that different controls are used for editing particular properties of an object.

Applicants respectfully disagree with and traverse such a rejection. Again, the present invention provides not only that the control is provided by the object itself, but the control is used to display and edit the property of the object itself in a list of properties, and the control is a custom control that defines a user interface for displaying and editing the property. As described above, Hirsch fails to provide such a custom control that defines a user interface for displaying and editing the property. Instead, Hirsch merely selects a control to use from an enumerated list of controls. There is not even a remote hint at defining a custom control. Further, the claimed custom control is used to display the property from the object itself and is not in the form of Hirsch's object inspector that is displaying a property from another object.

In view of the above, Applicants assert that the various elements of Applicants' claimed invention together provide operational advantages over the systems disclosed in Hirsch. In addition, Applicants' invention solves problems not recognized by Hirsch.

Thus, Applicants submit that independent claims 1, 5, 8, 13, and 17 are allowable over Hirsch. Further, dependent claims 2-4, 6-7, 9-12, 14-16, and 18 are submitted to be allowable over Hirsch in the same manner, because they are dependent on independent claims 1, 5, 8, 13, and 17, respectively, and because they contain all the limitations of the independent claims. In addition, dependent claims 2-4, 6-7, 9-12, 14-16, and 18 recite additional novel elements not shown by Hirsch.

V. CONCLUSION

In view of the above, it is submitted that this application is now in good order for allowance and such allowance is respectfully solicited. Should the Examiner believe minor matters still remain that can be resolved in a telephone interview, the Examiner is urged to call Applicants' undersigned attorney.

The Director is authorized to charge any additional fee(s) or any underpayment of fee(s), or to credit any overpayments to Deposit Account Number 50-0494 of Gates & Cooper LLP. Please ensure that Attorney Docket Number 30566.296-US-U1 is referred to when charging any payments or credits for this case.

Respectfully submitted,

John G. Beltran et al.

By their attorneys,

GATES & COOPER LLP

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: March 24, 2008

By: /Jason S. Feldmar/
Name: Jason S. Feldmar
Reg. No.: 39,187